	Application Name	FOXY FUNCTIONS	Ver.:	3.0
	Document Title	APPLICATION GUIDE	Rev.:	1.0
	Author	Vlad Feldfix	Date:	2023-01-16

FOXY FUNCTIONS v3.0

By Vlad Feldfix

Program Description

Foxy Functions is a Python library to help standardizes other applications by PIXEL FOX Games and Apps.

Using Instructions:

1. Download FoxyFunctions.py from Git Hub
2. Place FoxyFunctions.py in C:\...\Python\Python311\Lib
3. Start your app with: `from FoxyFunctions import ff`

Functions Descriptions:

`constructor(program_name, rev, main_menu)`

`program_name`: your app name

`rev`: your app version

`main_menu`: main menu object in the following format - (("LABEL", function)). e.g. (("RUN", self.run), ("READ SCRIPT", self.read))

Notice! that there are 3 default functions SETTINGS, HELP, EXIT

`run()`

Place this at the end of your app's __init__ function

`write(text)`

`text`: the text that you want to export to the console

`clear()`

Clears the console. no arguments needed

`msg(text)`

`text`: the content of the message

`error(text)`

`text`: the content of the error

`question(text)`

`text`: the content of the question

- Returns a "yes" or "no" value


`settings_get(variable)`

`variable`: returns the given variable

`progress_bar_value_set(percent)`

`percent`: a number int or float between 0-100

`progress_bar_value_get()`

	Application Name	FOXY FUNCTIONS	Ver.:	3.0
	Document Title	APPLICATION GUIDE	Rev.:	1.0
	Author	Vlad Feldfix	Date:	2023-01-16

- Returns the current number of the progress bar

`form(form_title, variables, on_submit)`

form_title: the title on top of the form. e.g. Add new user

variables: are the form fields. e.g. ("Name", "Email", "Password")

each variable can also be a Tuple where

index 0 is the field name, 1 is a default value, 2 is DISABLE EDIT True or False, and 3 is Cannot be empty True or False.

e.g. (("ID", userid, False), ("Name", "No name"), ("Email", "No Email", True, False), "Password")

on_submit: what happens when you submit the form? e.g. self.on_submit

Notice! the submit function is getting 1 argument called data, this argument is a dictionary of the form results

`database_display(db, table, addfunction, editfunction, deletefunction)`

db: database full address and name. e.g. "D:\Users\Projects\sample_database.db"

table: which database table you want to see / edit

addfunction: what function to call when the Add button is pressed

editfunction: what function to call when the Edit button is pressed (gets the row to edit as a dictionary argument called data)

deletefunction: what function to call when the Delete button is pressed (gets the row to delete as a dictionary argument called data)

`update_database_display()`

updates the database display. no arguments needed

`read_script(scriptfilename)`

scriptfilename: script location

- Returns a list of all the script lines

`today()`

- Returns datestamp string YYYY-MM-DD

`csv_to_list(filename)`

filename: the location of the .csv file you want to make into a list

- Returns a list of lists in the following format:

[[row1col1, row1col2, row1col3, ...], [row2col1, row2col2, row2col3, ...], [row3col1, row3col2, row3col3, ...] ...]